Theory of Automata and Languages

Pushdown Automata and Context-Free Languages - 2

Fall 2024

Sharif University of Technology

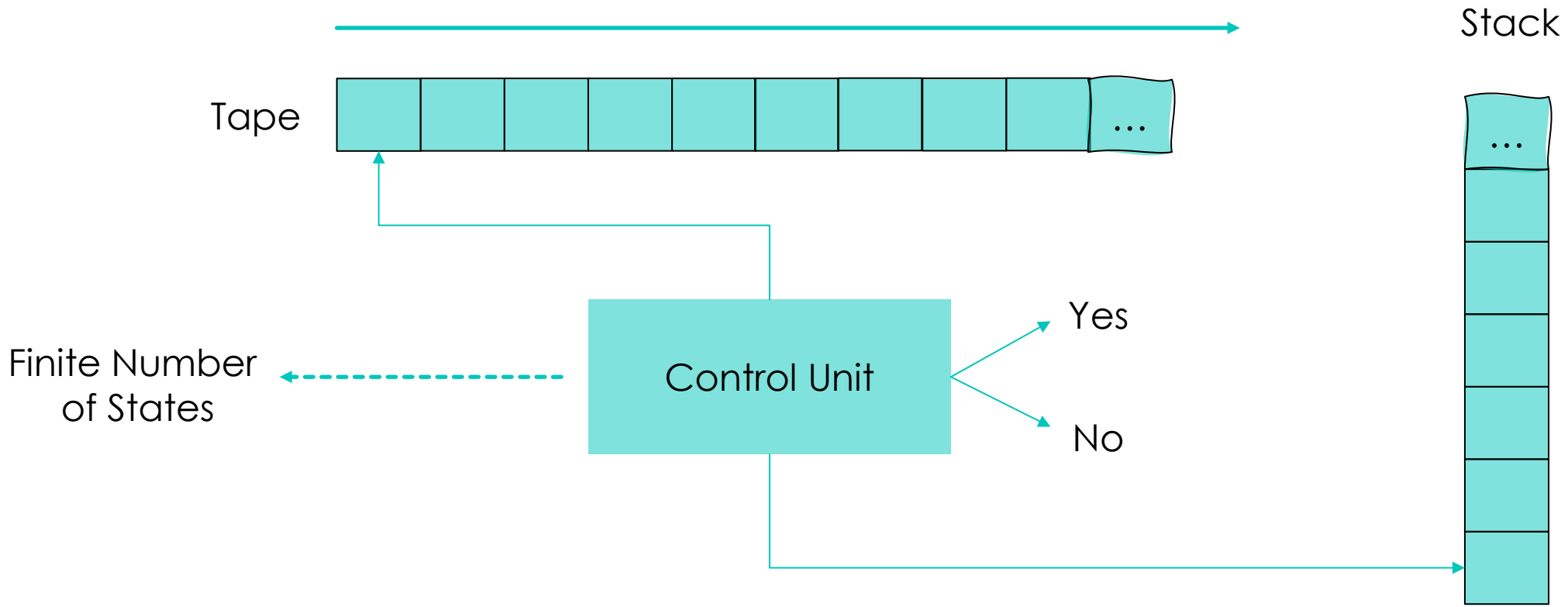
Mehran Moeini Jam



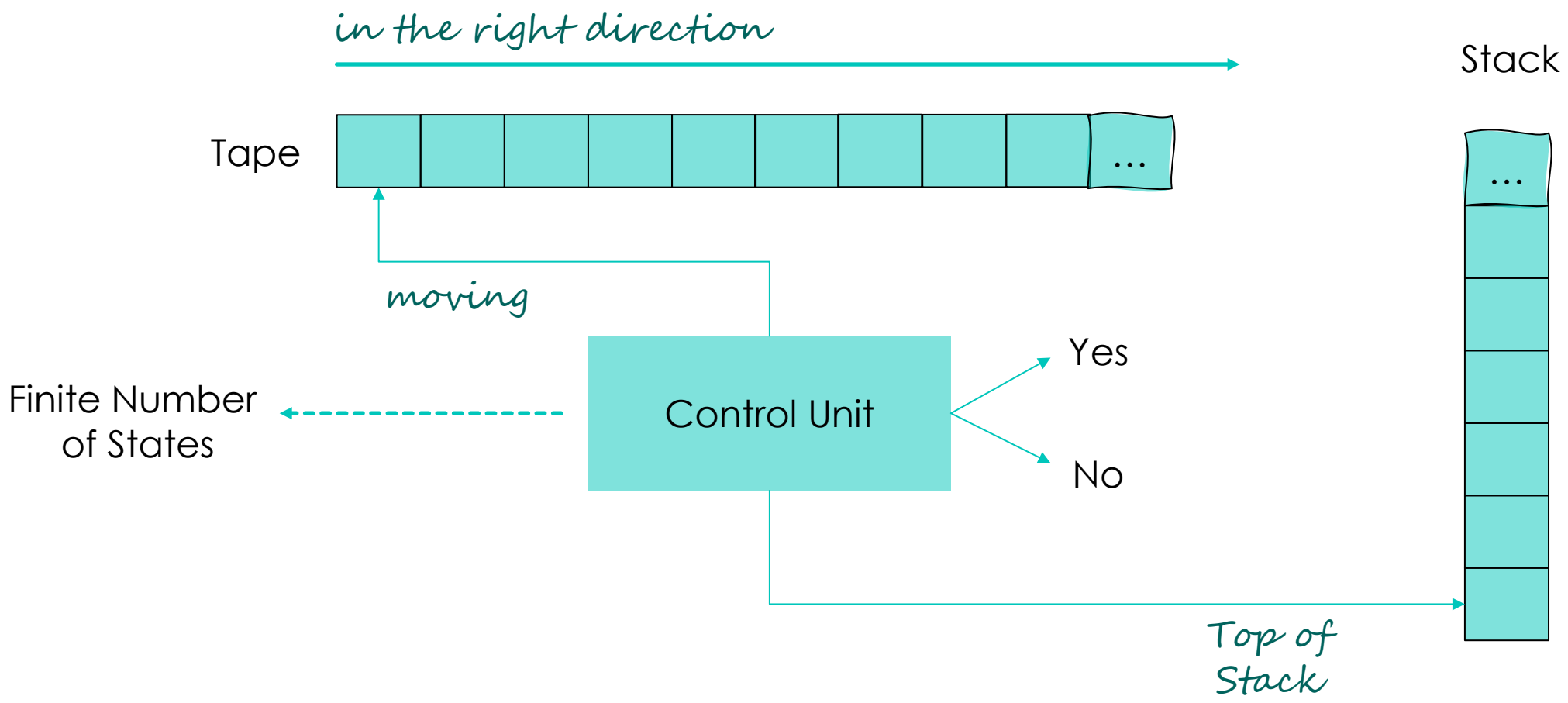
Overview

- Non-deterministic Pushdown Automata (NPDA)
- String and Language Recognition
- Instantaneous Description
- PDAs and CFGs Equivalence
- The Pumping Lemma for Context-free Languages
- Closure Properties Regarding CFLs
- Deterministic Pushdown Automata (DPDA)

Pushdown Automata (as an acceptor)



Pushdown Automata (as an acceptor)



Non-Deterministic Pushdown Automata (NPDA)

A Non-Deterministic Pushdown Automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called states,
- Σ is a finite set called the input alphabet,
- Γ is a finite set called the stack alphabet,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow 2^{Q \times \Gamma_\varepsilon}$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accept (final) states.

String Recognition

- A Non-Deterministic Pushdown Automata accepts a string $w = w_1 w_2 \ldots w_m$

(where each w_i is a member of Σ_ε) when a sequence of states $r_0, r_1, \ldots, r_m$ in Q and

strings $s_0, s_1, \ldots, s_m \in \Gamma_\varepsilon^*$ exist, such that :

1. $r_0 = q_0$ $\quad$ $s_0 = \varepsilon$
2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ $\quad$ for $i = 0, \ldots, m - 1$

   $s_i = at$ $\quad$ $s_{i+1} = bt$ $\quad$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
3. $r_m \in F$

String Recognition

- A Non-Deterministic Pushdown Automata accepts a string $w = w_1 w_2 \ldots w_m$

(where each w_i is a member of Σ_ε) when a sequence of states $r_0, r_1, \ldots, r_m$ in Q and strings $s_0, s_1, \ldots, s_m \in \Gamma_\varepsilon^*$ exist, such that :

we are embedding the "epsilon-transitions" into the string itself

1. $r_0 = q_0 \qquad s_0 = \varepsilon$
2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a) \qquad$ for $i = 0, \ldots, m - 1$

   $s_i = at \qquad s_{i+1} = bt \qquad$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
3. $r_m \in F$

Instantaneous Description

The accepted language can also be summarized in instantaneous description format:

$$L(P) = \{ w \mid (q_0, w, \varepsilon) \overset{*}{\vdash} (p, \varepsilon, \gamma) \text{ for some } p \text{ in } F \text{ and } \gamma \text{ in } \Gamma^* \}$$

In each triple, the first element represents the current state, the second element represents the unprocessed part of the input string, and the third element represents the contents of the stack. The $\vdash$ represents a single step in the corresponding PDA and the $\overset{*}{\vdash}$ is defined accordingly for zero or more steps.

String Recognition

- By the previous definition, an input string is effectively accepted when the automaton reads the input entirely, and reaches a final state at the end, and along the way, it pushes and pops symbols into/from the stack according to the defined transition function.

- This definition doesn't require the stack to become empty at the end, although the emptiness is usually considered as a desired condition.

String Recognition – Extended Definition

- For every NPDA with the accepting condition defined as "reaching a final state on input end", an equivalent NPDA can be defined with the accepting condition defined to be "reaching a final state on input end and the stack being empty".

- For this purpose, a new NPDA $P' = (Q \cup \{q_f\}, \Sigma, \Gamma, \delta' , q_0, \{q_f\})$ is constructed, where the transition function δ' is formed from δ augemented with the following new transitions:

$\delta'(q_i, \varepsilon, \varepsilon) = \{ (q_f, \varepsilon) \}$ for all $q_i \in F$

$\delta'(q_f, \varepsilon, A) = \{ (q_f, \varepsilon) \}$ for all $A \in \Gamma$

Contract

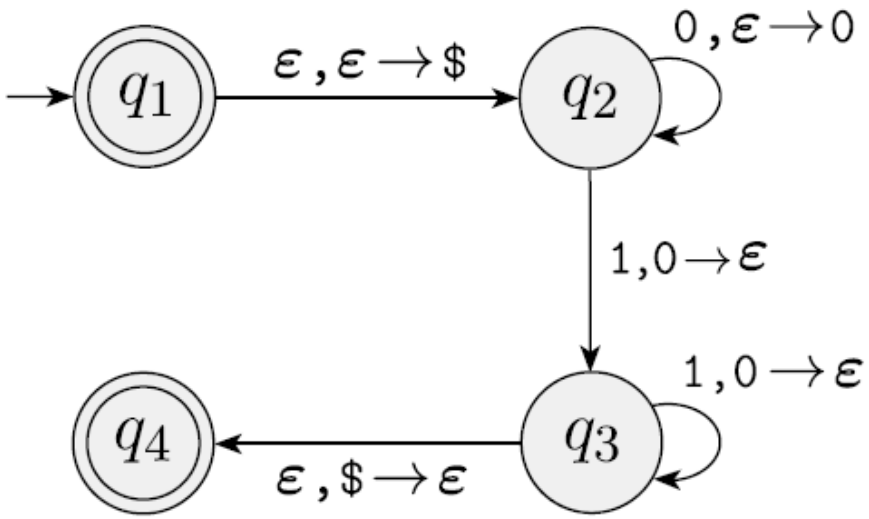
Henceforward, we consider the accepting condition to be the latter definition

Example

- Consider the language $L = \{0^n1^n \mid n \geq 0\}$

Example

- Consider the language $L = \{0^n1^n \mid n \geq 0\}$

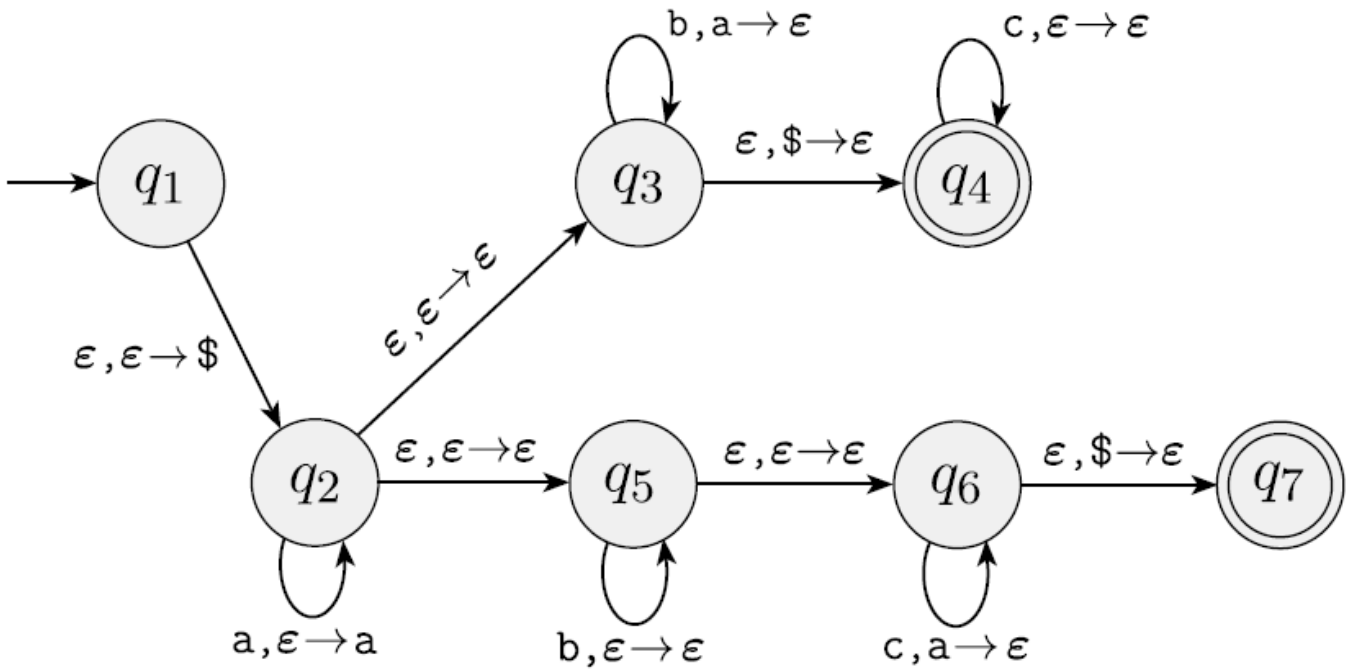


Example

- Consider the language $L = \{ a^i b^j c^k \mid i,j,k \geq 0 \text{ and } (i = j \text{ or } i = k) \}$

Example

- Consider the language L = { $a^i b^j c^k$ | i,j,k ≥ 0 and (i = j or i = k) }

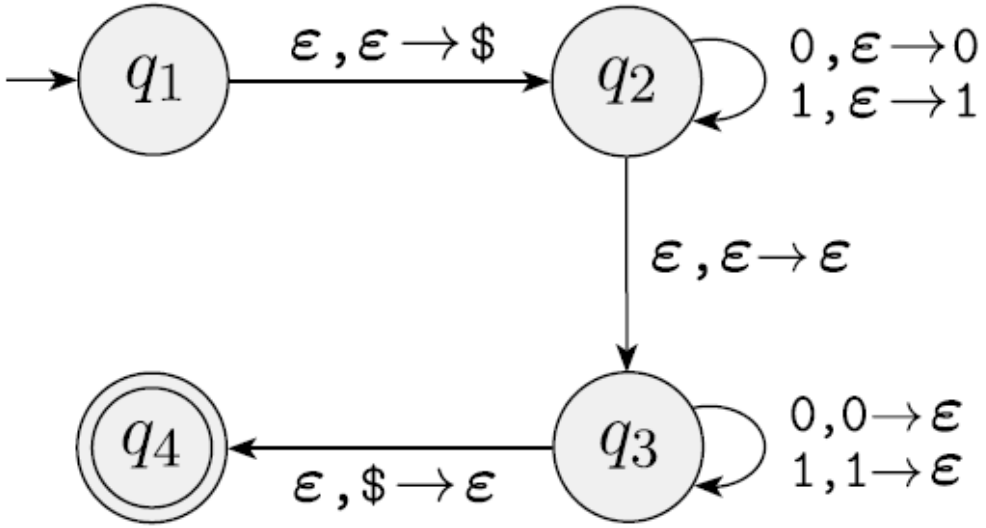


Example

- Consider the language $L = \{ ww^R \mid w \in \{0,1\}^* \}$

Example

- Consider the language $L = \{ ww^R \mid w \in \{0,1\}^* \}$



PDAs and CFGs Equivalence – Part 1

- Theorem 1 : For every context-free grammar G, we can design a pushdown automaton P that accepts the same language that the grammar generates.

- The PDA P that will be described next, will work by accepting its input w, if G generates that input, by determining whether there is a derivation for w.

PDAs and CFGs Equivalence – Part 1, Proof Idea

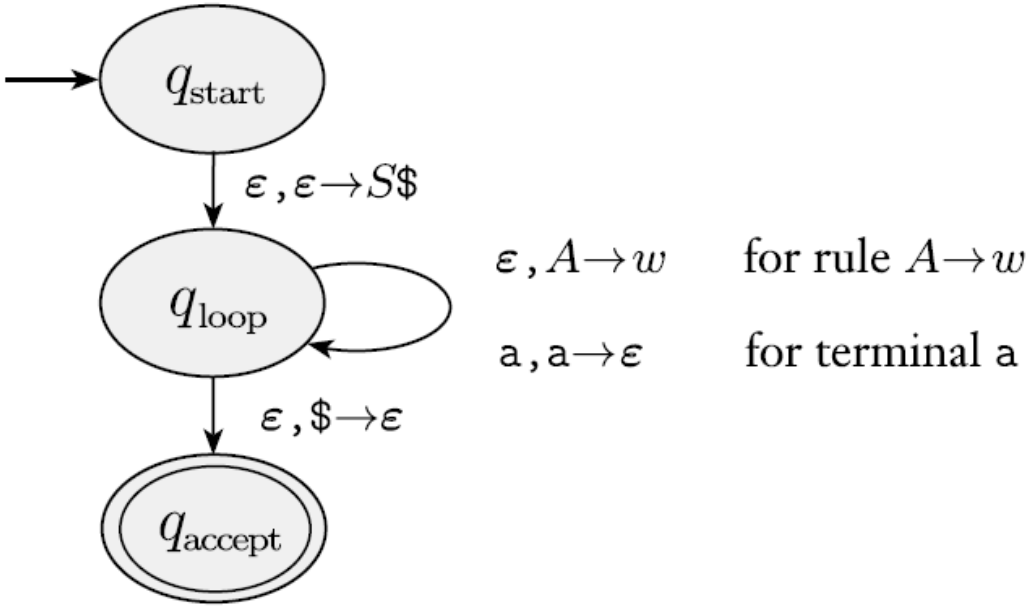
1. Place the marker symbol $ and the start variable on the stack.

2. Repeat the following steps forever.

- If the top of stack is a variable symbol A, non-deterministically select one of the rules for A and substitute A by the string on the right-hand side of the rule.
- If the top of stack is a terminal symbol a, read the next symbol from the input and compare it to a. If they match, repeat. If they do not match, reject on this branch of the nondeterminism.
- c. If the top of stack is the symbol $, enter the accept state. Doing so accepts the input if it has all been read.

PDAs and CFGs Equivalence – Part 1, Proof Idea

- The following is a general depiction of the corresponding PDA



Example

Consider the following grammar:

$$S \rightarrow aTb \mid b$$

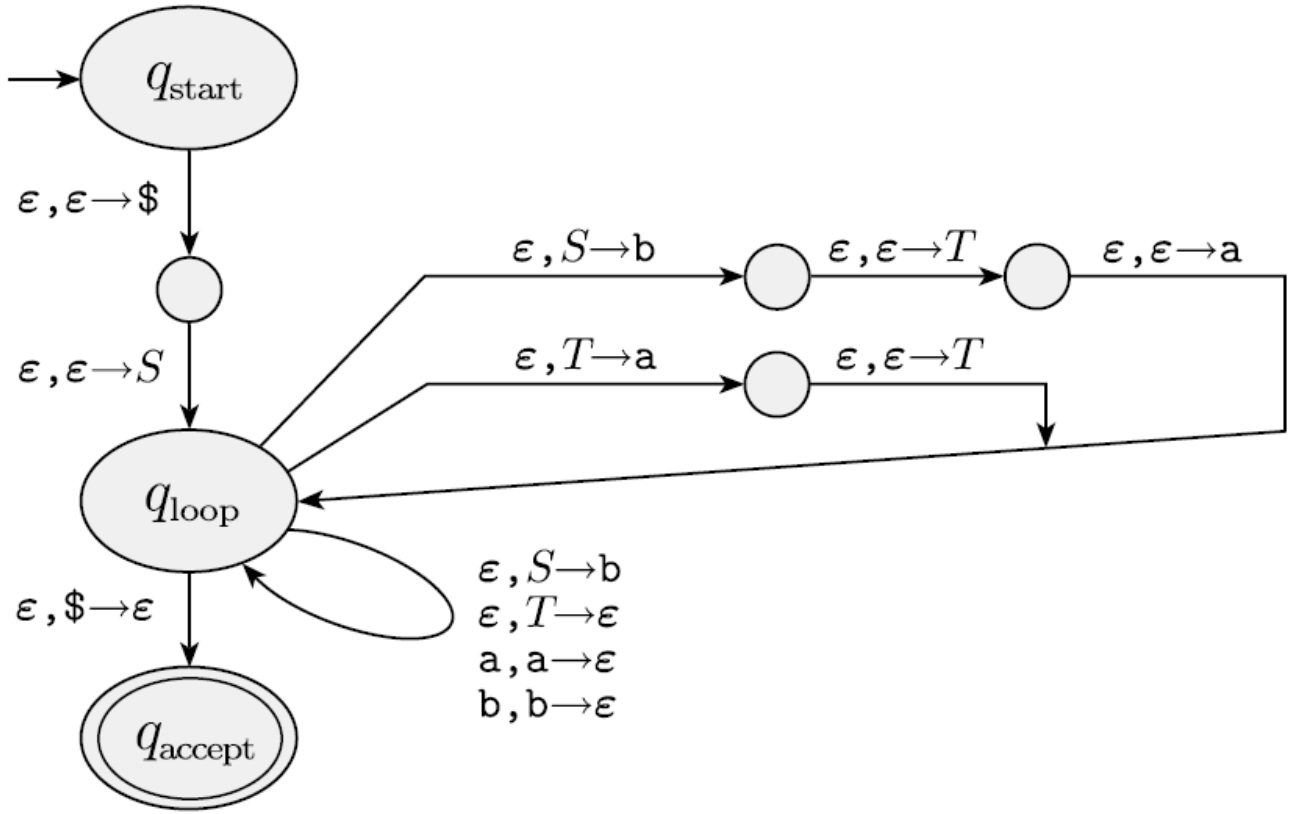
$$T \rightarrow Ta \mid \varepsilon$$

Example

Consider the following grammar:

$S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \varepsilon$

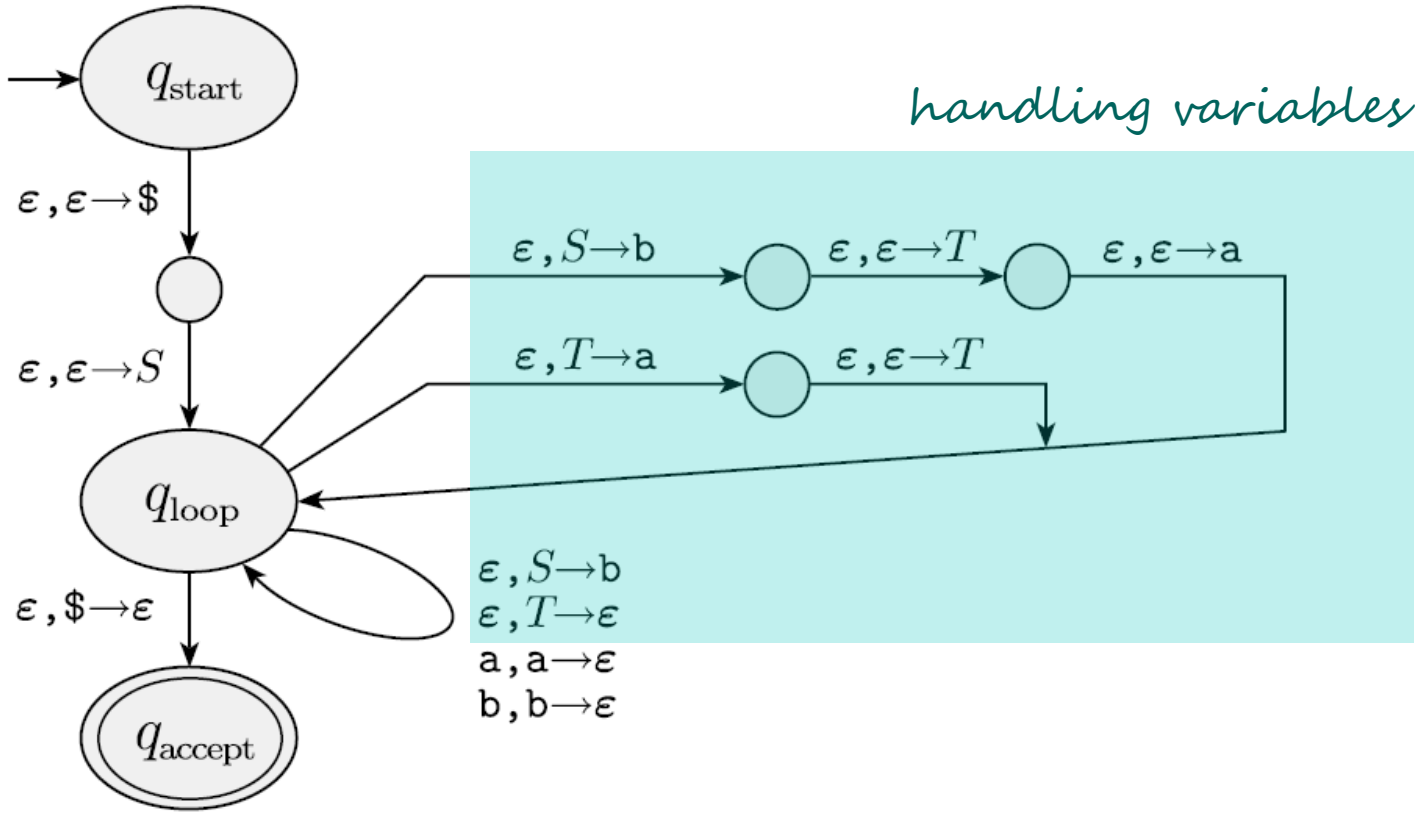


Example

Consider the following grammar:

$S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \varepsilon$

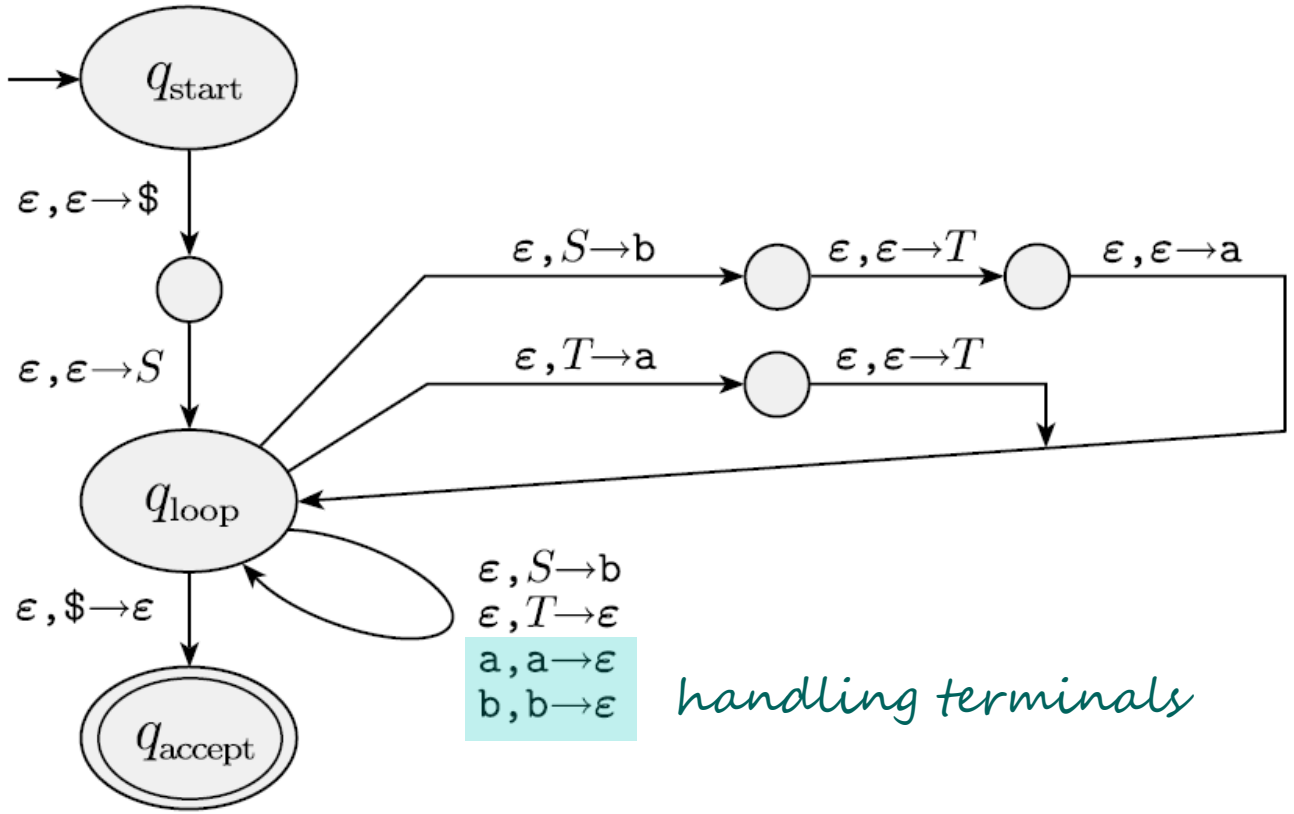


Example

Consider the following grammar:

S → aTb | b

T → Ta | ε



handling terminals

PDAs and CFGs Equivalence – Part 2

- Theorem 2: For every pushdown automaton P, we can design a context-free grammar G, that generates the same language that the automaton accepts.

- G should generate a string if that string causes the PDA to go from its start state to an accept state by reading it entirely.

PDAs and CFGs Equivalence – Part 2, Proof Idea

- First, we simplify our task by modifying P slightly to give it the following three features :

  1. It has a single accept state, q_{accept}
  2. It empties its stack before accepting.
  3. Each transition either pushes a symbol onto the stack (a push move) or pops one off the stack (a pop move), but it does not do both at the same time.

PDAs and CFGs Equivalence – Part 2, Proof Idea

- For each pair of states p and q in P, the grammar will have a variable A_{pq}
- This variable generates all the strings that can take P from p with an empty stack to q with an empty stack.
- The start variable then will be $A_{q_0, q_{accept}}$

PDAs and CFGs Equivalence – Part 2, Proof Idea

- Two possibilities occur during P's computation on each string x

1. The symbol that popped at the end is the same symbol that was pushed at the beginning.

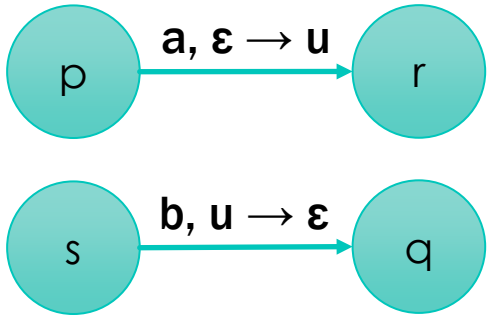
2. The symbol that popped at the end isn't the same symbol that was pushed at the beginning.

PDAs and CFGs Equivalence – Part 2, Proof Idea

- Consider $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$
- The variables of G(L(P)) are $\{A_{pq} \mid p, q \in Q\}$ and The start variable is $A_{q_0, q_{accept}}$

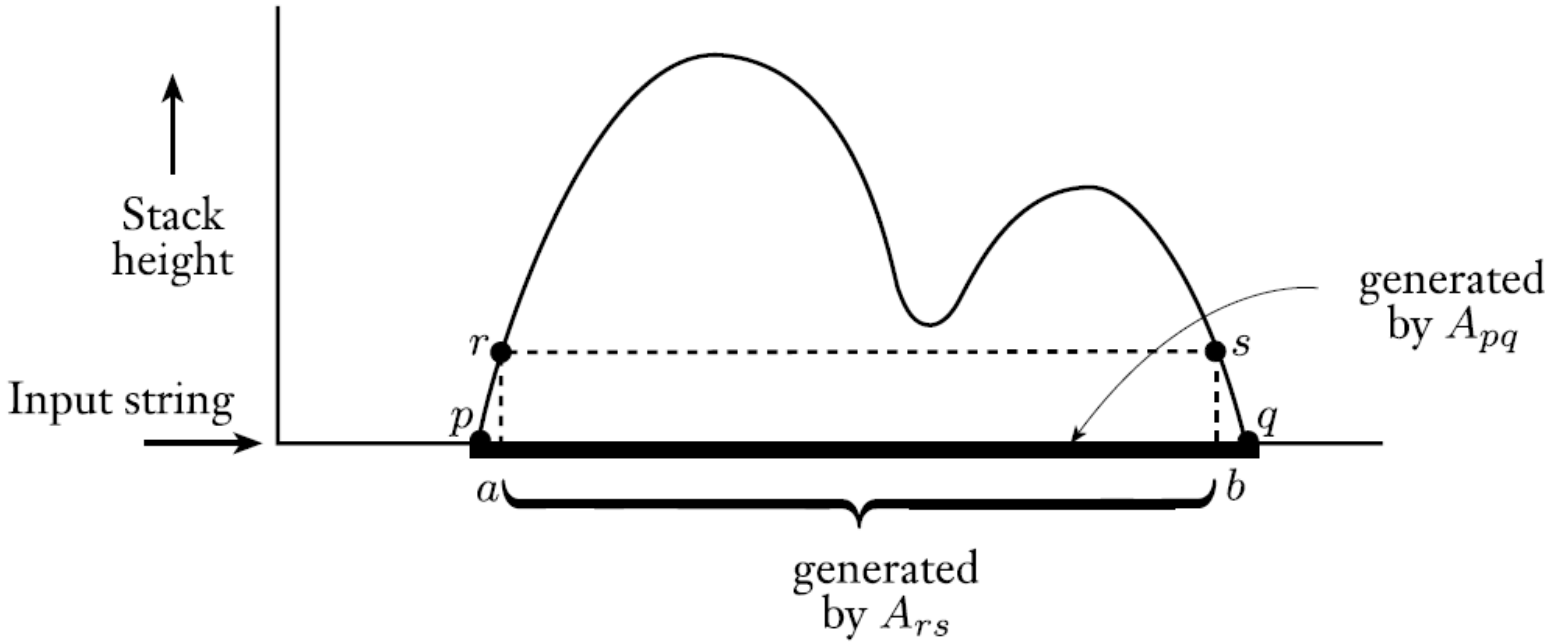
1. The symbol that popped at the end is the same symbol that was pushed at the beginning.

For each $p, q, r, s \in Q$, $u \in \Gamma$ and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, u), and $\delta(s, b, u)$ contains (q, ε), put the rule $A_{pq} \rightarrow aA_{rs}b$ in G.



PDAs and CFGs Equivalence – Part 2, Proof Idea

For each $p, q, r, s \in Q$, $u \in \Gamma$ and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, u), and $\delta(s, b, u)$ contains (q, ε), put the rule $A_{pq} \rightarrow aA_{rs}b$ in G.



PDAs and CFGs Equivalence – Part 2, Proof Idea

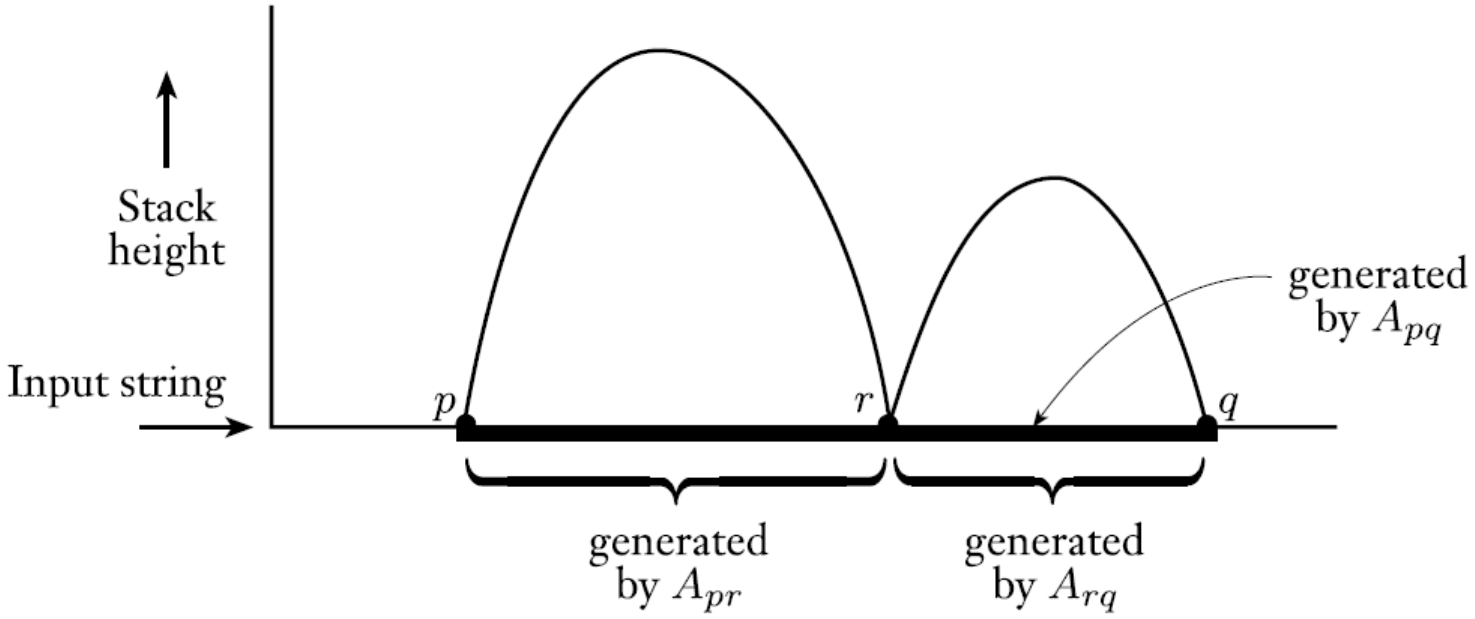
- Consider $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$
- The variables of $G(L(P))$ are $\{A_{pq} \mid p, q \in Q\}$ and The start variable is $A_{q_0, q_{accept}}$

2. The symbol that popped at the end isn't the same symbol that was pushed at the beginning.

For each $p, q, r \in Q$, put the rule $A_{pq} \rightarrow A_{pr}A_{rq}$ in G.

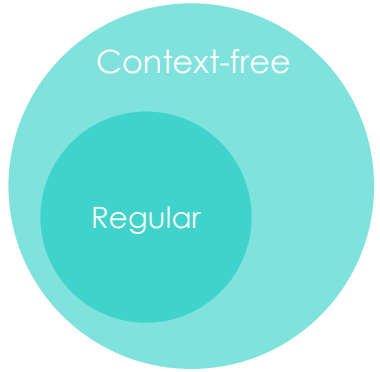
PDAs and CFGs Equivalence – Part 2, Proof Idea

For each p, q, r ∈ Q, put the rule $A_{pq} \rightarrow A_{pr}A_{rq}$ in G.



Relation with Regular languages

- Every regular language is also a context-free language. So the class of regular languages is a proper subset of the class of context-free languages.



- To prove the above statement, note that every DFA can be depicted as a NPDA that doesn't push anything on the stack. So the transitions become of the form

  $a, \varepsilon \rightarrow \varepsilon$ where a is any symbol in the input alphabet.

Review: The Pumping Lemma for Regular Languages

- The formal definition is as follows.

  Consider **Regular** to be the class of regular languages.

$$(\forall\, L \in \text{Regular})\ (\exists\, n \in \mathbb{N})(\forall\, \omega \in L),$$

$$\big(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\, x, y, z \in \Sigma^*),\ (\omega = xyz,\ |xy| \leq n,\ |y| \geq 1,\ (\forall i \in \mathbb{N}),\ (xy^i z \in L)\ )\ )\ \big)$$

The Pumping Lemma for Context-free Languages

- The formal definition is as follows.
- Consider **CF** to be the class of context-free languages

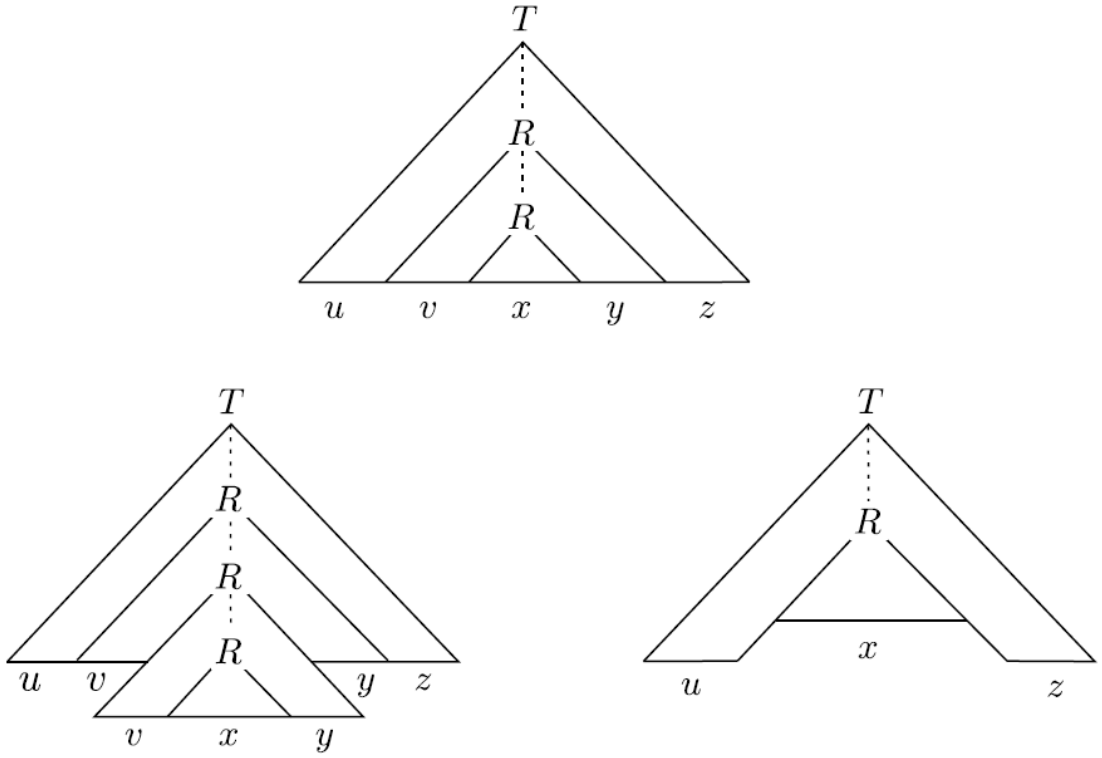
$$(\forall\, L \in CF)\ (\exists\, n \in \mathbb{N})(\forall\, \omega \in L),$$

$$((\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ u,v,x,y,z \in \Sigma^*),\ (\omega = uvxyz,\ |vxy| \leq n,\ |vy| \geq 1,\ (\forall i \in \mathbb{N}),\ (uv^ixy^iz \in L)\ )\ )\ )$$

The Pumping Lemma for Context-free Languages

$$(\forall\, L \in CF)\ (\exists\, n \in \mathbb{N})(\forall\, \omega \in L),$$

$$(\ (\ |\omega| \geq n\ ) \Rightarrow (\ (\exists\ u,v,x,y,z \in \Sigma^*),\ (\omega = uvxyz,\ |vxy| \leq n,\ |vy| \geq 1,\ (\forall i \in \mathbb{N}),\ (uv^ixy^iz \in L)\ )\ )\ )$$



Example

- $L = \{a^k b^k c^k \mid k \geq 0\}$

- Consider $\omega = a^n b^n c^n$ as the string, then $|\omega| = 3n$
- Let i = 0, the new string ω' becomes uxz for any given symbols
- vy can't consist all three symbols, and uxz can not be in the proper form (check!) ※

$$(\forall\, L \in CF)\ (\exists\, n \in \mathbb{N})(\forall\, \omega \in L),$$

$$((\,|\omega| \geq n\,) \Rightarrow (\,(\exists\, u,v,x,y,z \in \Sigma^*), (\omega = uvxyz,\ |vxy| \leq n,\ |vy| \geq 1, (\forall i \in \mathbb{N}), (uv^ixy^iz \in L)\,)\,)\,)$$

Operations on CFLs : Union

- Let L_1 and L_2 be context-free languages generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and

  $G_2 = (V_2, \Sigma_2, R_2, S_2)$ respectively, then define the grammar G for $L_1 \cup L_2$ as follows:

$$G = (V_1 \cup V_2 \cup \{S_0\}\ ,\ \Sigma_1 \cup \Sigma_2\ ,\ R_1 \cup R_2 \cup \{\ S_0 \rightarrow S_1 | S_2\ \}\ ,\ S_0)$$

- So the class of context-free languages is closed under the union operation.

Operations on CFLs : Concatenation

- Let L_1 and L_2 be context-free languages generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and

  $G_2 = (V_2, \Sigma_2, R_2, S_2)$ respectively, then define the grammar G for $L_1 . L_2$ as follows:

$$G = (V_1 \cup V_2 \cup \{S_0\} , \Sigma_1 \cup \Sigma_2 , R_1 \cup R_2 \cup \{ S_0 \rightarrow S_1 S_2 \} , S_0)$$

- So the class of context-free languages is closed under the concatenation operation.

Operations on CFLs : Kleene Star

- Let L be context-free languages generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$, then define the grammar G for L_1^* as follows:

$$G = (V_1 \cup \{S_0\}, \Sigma_1, R_1 \cup \{ S_0 \rightarrow S_1 S_0 \mid \varepsilon \}, S_0)$$

- So the class of context-free languages is closed under the kleene star operation.

Operations on CFLs : Intersection

- Let
- $L_1 = \{a^k b^k c^j \mid k, j \geq 0\}$
- $L_2 = \{a^j b^k c^k \mid k, j \geq 0\}$
- The language $L = L_1 \cap L_2$ is $\{a^k b^k c^k \mid k \geq 0\}$ which was proven to be non-context-free.
- So the class of context-free languages is not closed under the intersection operation.

Operations on CFLs : Complement

- Let L_1 and L_2 be any two context-free languages.

- Consider the class closed under the complement operator, then the language

  $L = (L_1^c \cup L_2^c)^c = L_1 \cap L_2$ must also be context-free, which is not necerssarily true,

  and so it's a contradiction. ※

- So the class of context-free languages is not closed under the complement operation.

Deterministic Pushdown Automata (DPDA)

A Deterministic Pushdown Automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called states,
- Σ is the finite input alphabet and Γ is the finite stack alphabet,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow (Q \times \Gamma_\varepsilon) \cup \{\emptyset\}$ is the transition function,

  where for every $q \in Q$, $a \in \Sigma$, $x \in \Gamma$, exactly one of the following is not $\emptyset$:

  $\delta(q, a, x)$, $\delta(q, a, \varepsilon)$, $\delta(q, \varepsilon, x)$, $\delta(q, \varepsilon, \varepsilon)$

- $q_0 \in Q$ is the start state, and $F \subseteq Q$ is the set of accept (final) states.

Deterministic Pushdown Automata (DPDA)

A Deterministic Pushdown Automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called states,
- Σ is the finite input alphabet and Γ is the finite stack alphabet,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow (Q \times \Gamma_\varepsilon) \cup \{\emptyset\}$ is the transition function,

  where for every $q \in Q$, $a \in \Sigma$, $x \in \Gamma$, exactly one of the following is not $\emptyset$:

non-empty non-empty non-empty empty empty non-empty empty empty

$\delta(q, a, x)$, $\delta(q, a, \varepsilon)$, $\delta(q, \varepsilon, x)$, $\delta(q, \varepsilon, \varepsilon)$

- $q_0 \in Q$ is the start state, and $F \subseteq Q$ is the set of accept (final) states.

Deterministic Pushdown Automata (DPDA)

- The class of all languages that can be accepted by a deterministic pushdown automaton is called the class of deterministic context-free languages.



- The class is a proper subset of the class of context-free languages and a proper superset of the class of regular languages.

- We won't further pursue this class of languages.

